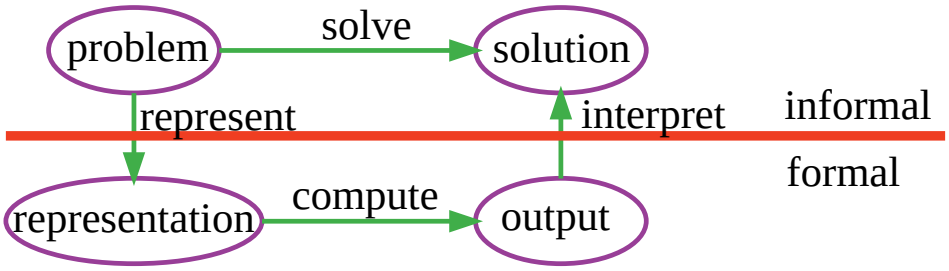
Representing Knowledge

Given a problem to solve, how do you solve it?

- What is a solution to the problem?
- What do you need in the language to represent the problem?
- How can you map from the informal problem description to a representation of the problem?
- What distinctions in the world are important to solve the problem?
- What knowledge is required?
- What level of detail is required?

- What reasoning strategies are appropriate?
- Is worst-case performance or average-case performance the critical time to minimize?
- Is it important for a human to understand how the answer was derived?
- How can you acquire the knowledge from experts or from experience?
- How can the knowledge be debugged, maintained, and improved?

Knowledge representation framework



Defining a Solution

- Given an informal description of a problem, you need to determine what would constitute a solution.
- Typically much is left unspecified, but the unspecified parts can't be filled in arbitrarily.
- Much work in AI is motivated by common-sense reasoning. You want the computer to be able to make common-sense conclusions about the unstated assumptions.

Quality of Solutions

Does it matter if the answer is wrong or answers are missing?

Classes of solution:

Optimal solution the best solution according some measure of solution quality.

Satisficing solution one that is good enough, according to some description of which solutions are adequate.

Approximately optimal solution one whose measure of quality is close to the best theoretically possible.

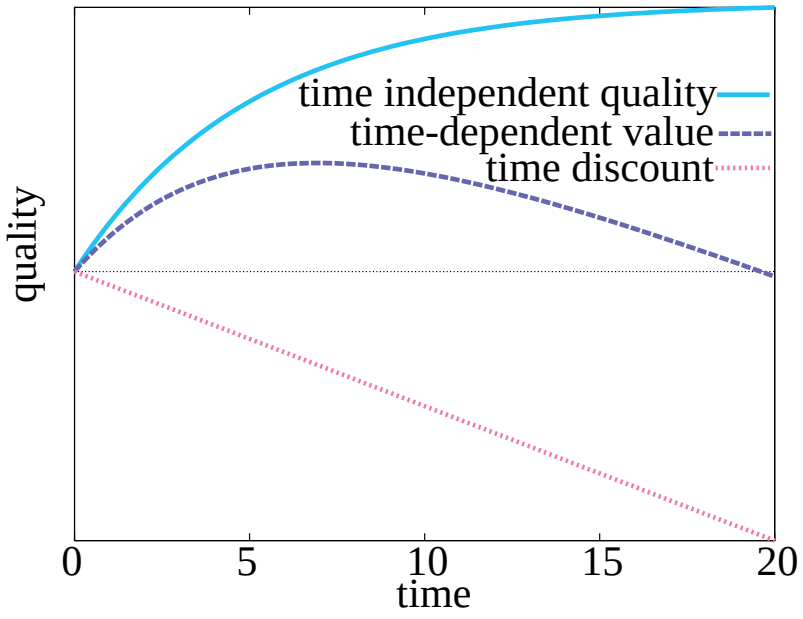
Probable solution one that is likely to be a solution.

Decisions and Outcomes

- Good decisions can have bad outcomes. Bad decisions can have good outcomes.
- Information can be valuable because it leads to better decisions: **value of information.**
- You have to trade off computation time and solution quality: an **anytime algorithm** can provide a solution at any time; given more time it can produce better solutions.

You don't only need to be concerned about finding the right answer, but about acquiring the appropriate information, and computing it in a timely manner.

Solution quality and computation time



Choosing a Representation Language

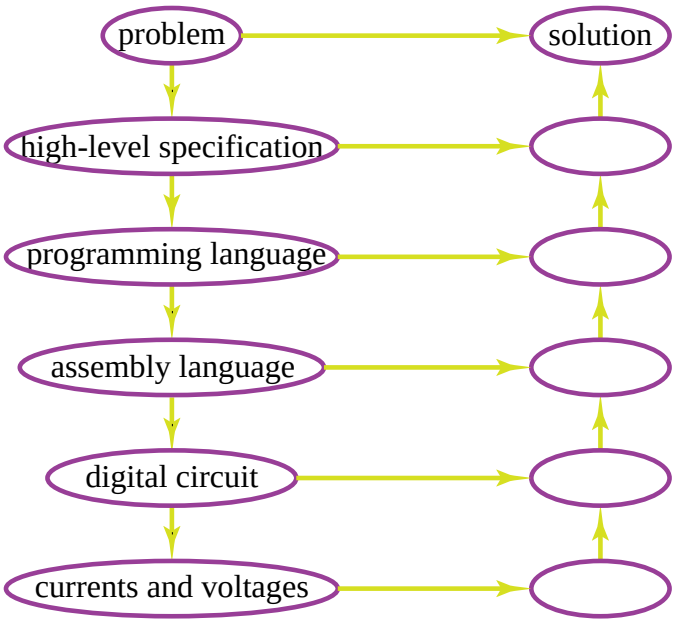
You need to represent a problem to solve it on a computer.

problem
⟶ specification of problem
⟶ appropriate computation

Example representations: C++, CILog/Prolog, English

A logic is a language + specification of what follows from input in that language.

Hierarchy of representations



Knowledge & Symbol Levels

Two levels of abstraction seem to be common among biological and computational entities:

Knowledge level in terms of what an agent knows and what an agent's goals are

Symbol level in terms of what symbols the agent is manipulating.

The knowledge level is about the external world to the agent.

The symbol level is about what symbols an agent uses to implement the knowledge level.

Mapping from Problem to Representation

- What level of abstraction of the problem do you want to have to represent?
- What objects and relations in the world do you want to represent?
- How can you represent the knowledge to ensure that the representation is natural, modular, and maintainable?

Choosing a level of abstraction

- A high-level description is easier for a human to specify and understand.
- A low-level description can be more accurate and more predictive. High-level descriptions abstract away details that may be important for actually solving the problem.
- The lower the level, the more difficult it is to reason with.
- You may not know the information needed for a low-level description.

It is sometime possible to use multiple levels of abstraction.

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"
Can't ask "what is the color of pen_7?"

$color(pen_7, red)$. It's easy to ask "What's red?"
It's easy to ask "What is the color of pen_7?"
Can't ask "What property of pen_7 has value *red*?"

$prop(pen_7, color, red)$. It's easy to ask all these questions.

$prop(Object, Attribute, Value)$ is the only relation needed:
object-attribute-value representation

Universality of *prop*

To represent "a is a parcel"
$prop(a, is\_a, parcel)$, where *is_a* is a special relation
$prop(a, parcel, true)$, where *parcel* is a Boolean attribute

To represent $scheduled(cs422, 2, 1030, cc208)$. "section 2 of course *cs*422 is scheduled at 10:30 in room *cc*208." Let *b*123 name the booking:

$prop(b123, course, cs422)$.

$prop(b123, section, 2)$.

$prop(b123, time, 1030)$.

$prop(b123, room, cc208)$.

Semantics Networks

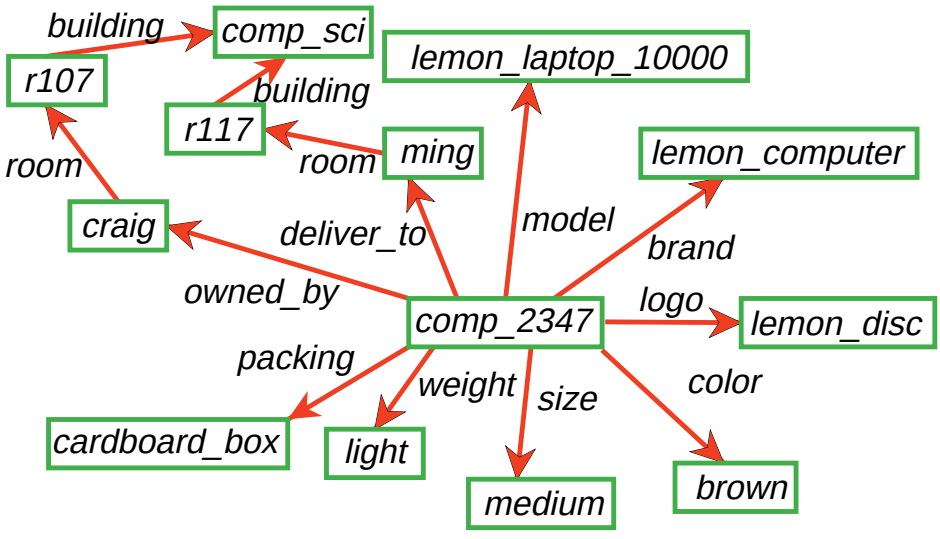
When you only have one relation, *prop*, it can be omitted without loss of information.

Write

$$prop(Obj, Att, Value)$$

as

An Example Semantic Network



Equivalent Logic Program

$prop(comp\_2347, owned\_by, craig).$

$prop(comp\_2347, deliver\_to, ming).$

$prop(comp\_2347, model, lemon\_laptop\_10000).$

$prop(comp\_2347, brand, lemon\_computer).$

$prop(comp\_2347, logo, lemon\_disc).$

$prop(comp\_2347, color, brown).$

$prop(craig, room, r107).$

$prop(r107, building, comp\_sci).$

$\vdots$

Frames

The properties and values for a single object can be grouped together into a frame.

We can write this as a list of *attribute* $=$ *value* or *slot* $=$ *filler*.

$[owned\_by = craig,$

$deliver\_to = ming,$

$model = lemon\_laptop\_10000,$

$brand = lemon\_computer,$

$logo = lemon\_disc,$

$color = brown,$

$\ldots]$

Primitive versus Derived Relations

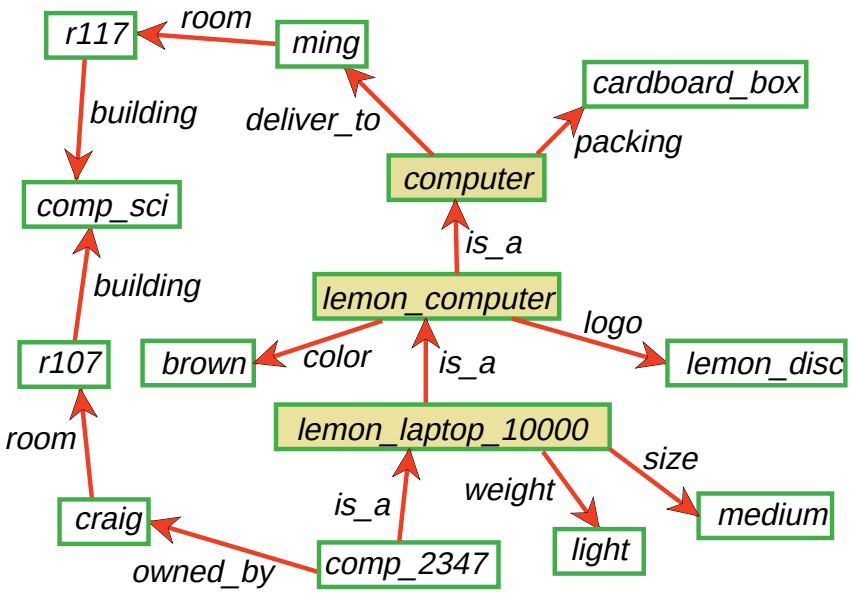
Primitive knowledge is that which is defined explicitly by facts.

Derived knowledge is knowledge defined by rules.

Example: All lemon laptops may have have *size* = *medium*. Associate this property with the class, not the individual.

Allow a special attribute *is_a* between an individual and a class or between two classes that allows for property inheritance.

A Structured Semantic Network



Logic of Property Inheritance

An arc $p \longrightarrow n$ from a class c means every individual in the class has value n of attribute p:

$$prop(Obj, p, n) \leftarrow$$
$$prop(Obj, is\_a, c).$$

Example:

$$prop(X, weight, light) \leftarrow$$
$$prop(X, is\_a, lemon\_laptop\_10000).$$
$$prop(X, is\_a, lemon\_computer) \leftarrow$$
$$prop(X, is\_a, lemon\_laptop\_10000).$$

Choosing Primitive and Derived Relations

- Associate an attribute value with the most general class with that attribute value.
- Don't associate contingent properties of a class with the class.
- Axiomatize in the causal direction. You want knowledge that is stable as the world changes.